

Tema 6: Segmentación y Paralelismo en el diseño de Computadores

Departament Arquitectura de Computadors

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya



Índice

- Introducción
- Paralelismo a Nivel de Instrucciones (ILP)
- Paralelismo a Nivel de Datos (DLP)
- Paralelismo a Nivel de Thread (TLP)
- Ejemplos Reales

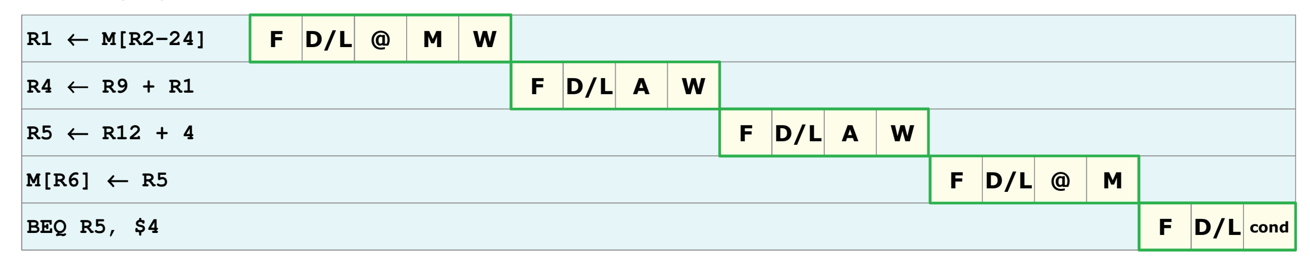
Introducción

Técnicas Básicas en el Diseño de Procesadores (¡ya vistas!):

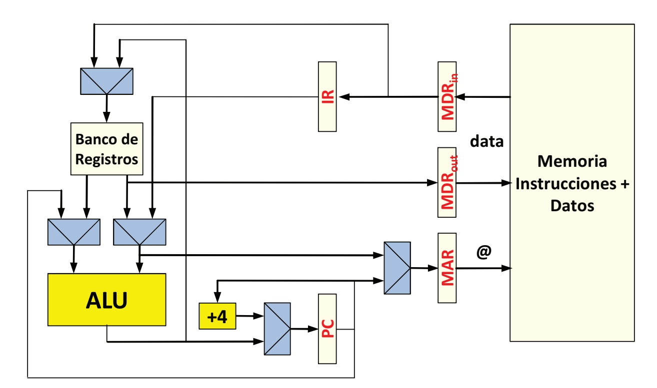
- **Memorización**
  - ✓ Memoria Cache
  - ✓ TLB
  - ✓ Predicción vía
- **Concurrencia**
  - Segmentación
    - ✓ Cache segmentada
    - ✓ Escrituras segmentadas
    - ✓ SDRAM
  - Paralelismo
    - ✓ Bancos DDR
    - ✓ Cache multibanco
    - ✓ RAIDs

Procesador Secuencial

- Las instrucciones se ejecutan de forma secuencial.
- Las instrucciones pueden tener tiempos de ejecución diferentes, dependiendo de su complejidad:



R1 ← M[R2−24]	F	D/L	@	M	W																	
R4 ← R9 + R1						F	D/L	A	W													
R5 ← R12 + 4										F	D/L	A	W									
M[R6] ← R5														F	D/L	@	M					
BEQ R5, $4																		F	D/L	cond		



Índice

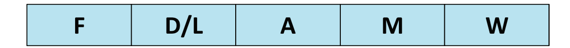
- **Introducción**
- **Paralelismo a Nivel de Instrucciones (ILP)**
  - Procesadores Segmentados
  - Superescalares
  - VLIW
- **Paralelismo a Nivel de Datos (DLP)**
- **Paralelismo a Nivel de Thread (TLP)**
- **Ejemplos Reales**

Procesadores Segmentados

- **Objetivo: ejecutar 1 instrucción por ciclo, CPI = 1.**
- Dificultades para alcanzar este objetivo
  - Los recursos hardware disponibles
  - Respetar la semántica del Lenguaje máquina.
- Para simplificar la segmentación se utiliza un LM tipo RISC

Aritméticas, lógicas y comparación	Rk ← Ri + Rj Rk ← Ri - Rj Rk ← Ri & Rj	Fetch	Decode	Read	ALU	Write	
Memoria	Rk ← M[Rj+despl]	Fetch	Decode	Read	@	MEM	Write
	M[Rj+despl] ← Rk	Fetch	Decode	Read	@	MEM	
Salto	BEQ Ri, $tag BNEQ Ri, $tag	Fetch	Decode	Read	Cond		

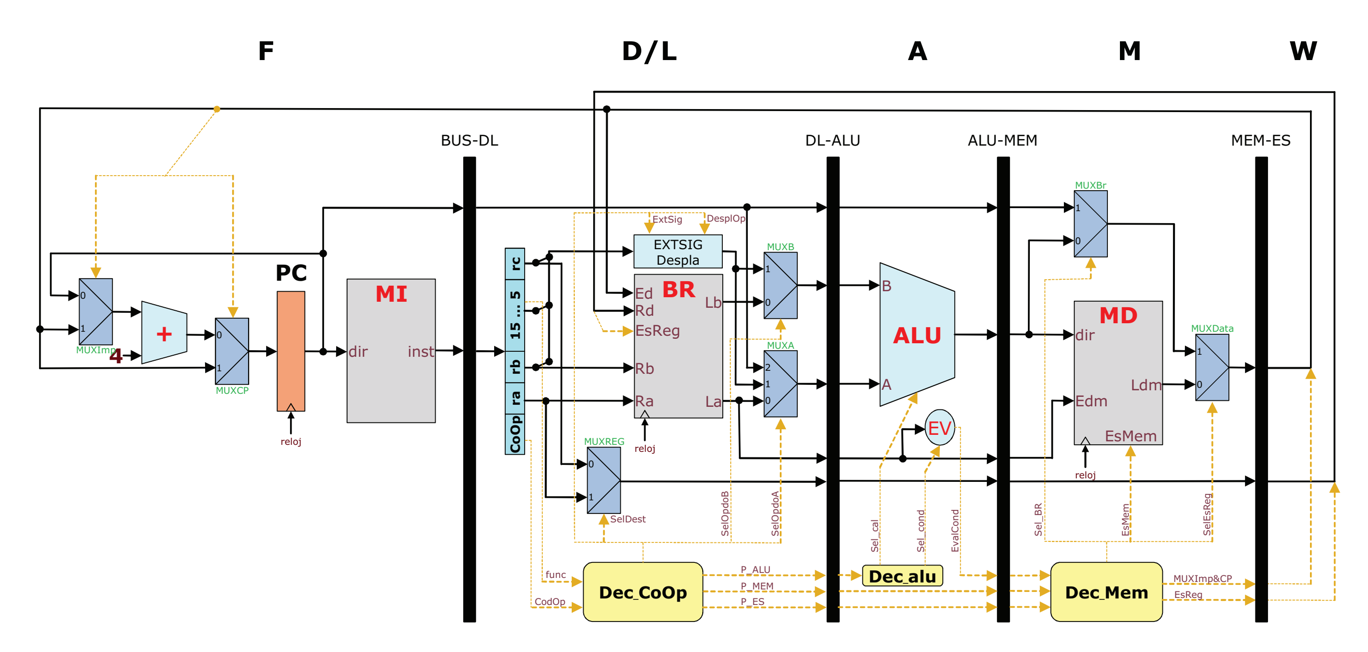
- Para simplificar el hardware, se busca que todas las instrucciones usen la misma segmentación:



F	D/L	A	M	W

Procesadores Segmentados

■ **Procesador segmentado en 5 etapas**



Todos los registros y elementos de almacenamiento se actualizan en el flanco ascendente de reloj
Para ver cuando se usa un elemento hay que observar sus conexiones, no su ubicación.



Procesadores Segmentados

Ejecución segmentada

	1	2	3	4	5	6	7	8	9	10
R1 ← M[R2-24]	F	D/L	A	M	W					
R4 ← R9 + R10		F	D/L	A	M	W				
R5 ← R12 + R11			F	D/L	A	M	W			
R6 ← R13 + R14				F	D/L	A	M	W		
R7 ← R18 + R19					F	D/L	A	M	W	

- Las instrucciones (datos + control) se mueven por el pipeline.
- Al inicio de ciclo, todas las etapas empiezan a funcionar con los datos que hay en los registros de desacoplo en la entrada de la etapa.
- Al final de ciclo, las señales de salida de cada etapa se almacena en los registros de desacoplo.
- En un ciclo determinado se está ejecutando una instrucción diferente en cada una de las etapas.
- **Prácticamente, con el hardware necesario para ejecutar 1 instrucción, estamos ejecutando concurrentemente 5 instrucciones independientes.**

Procesadores Segmentados

Límites a la segmentación

- **Riesgos de Datos**

	1	2	3	4	5	6	7
I_1: R1 ← M[R2+X]	F	D/L	A	M	W		
I_2: R4 ← R9 + R1		F	D/L	A	M	W	

La instrucción I_1 calcula R1 y es utilizado en la instrucción I_2. I_1 escribe R1 en el ciclo 5, I_2 lee el registro R1 en el ciclo 3 ⇒ **I_2 lee un valor incorrecto.**

- **Riesgos de Control**

	1	2	3	4	5	6	7
I_1: BEQ R1, $4	F	D/L	A	M	W		
I_2: R4 ← R9 + R10		F	D/L	A	M	W	
I_3: R5 ← R12 + R11			F	D/L	A	M	W

La instrucción I_1 evalúa la condición de salto en el ciclo 3 (etapa A). Hasta ese ciclo no sabemos si el salto será efectivo. Si el salto es efectivo ⇒ **Se ha iniciado la ejecución de 2 instrucciones erróneamente (I_2 e I_3).**

- **Riesgos Estructurales**
  - Se producen cuando un único recurso se intenta utilizar por 2 instrucciones diferentes.
  - El procesador segmentado que hemos presentado está libre de riesgos estructurales:
    - ✓ El Banco de Registros permite 2 lecturas y 1 escritura en el mismo ciclo
    - ✓ Dispone de Memorias independientes para instrucciones (MI) y datos (MD).

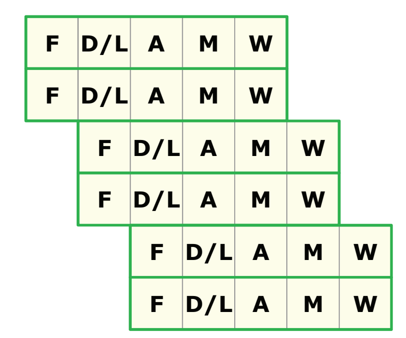
Procesadores Segmentados

Los procesadores son más complicados

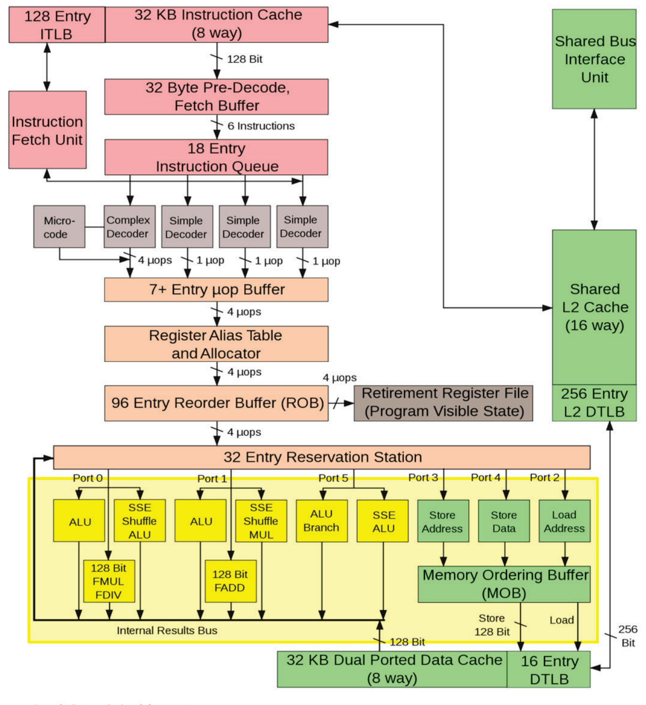
- **No todas las instrucciones han de tener las mismas etapas**
  - Operaciones complejas: multiplicación.
  - No siempre es fácil segmentar una operación: división.
  - Aritmética en Coma Flotante.
- **Instrucciones con tiempo de ejecución variable**
  - Accesos a Memoria: con fallo o acierto en cache (diferentes niveles).
- **Gestión de Interrupciones y Excepciones**

Procesadores Superescalares

- **Objetivo CPI < 1**
- Dispone de **múltiples Unidades Funcionales**
- Permite **iniciar más de una instrucción** (u operación) **por ciclo**
- Sigue siendo un procesador segmentado.
- Las instrucciones pueden tener tiempos de ejecución diferentes.
- La mayoría de los procesadores de propósito general actuales son superescalares.



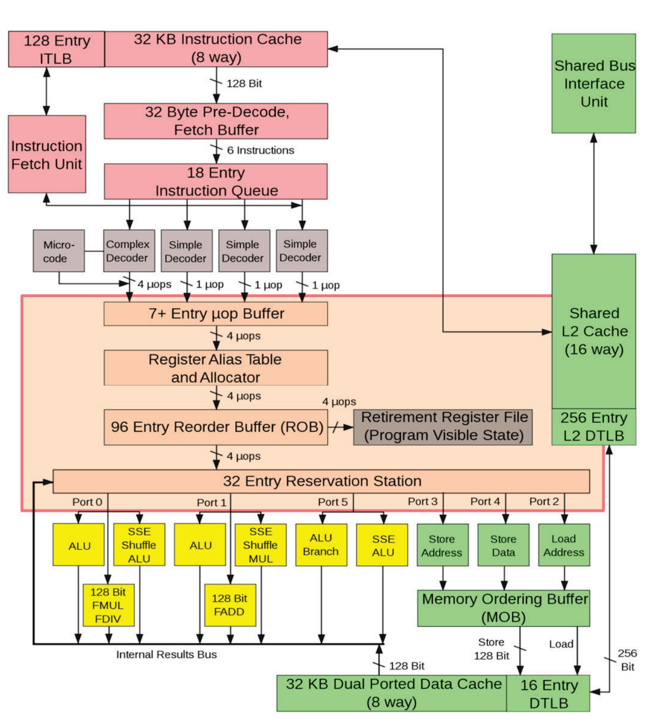
F	D/L	A	M	W		
F	D/L	A	M	W		
	F	D/L	A	M	W	
	F	D/L	A	M	W	
		F	D/L	A	M	W
		F	D/L	A	M	W



Intel Core 2 Architecture

Procesadores Superescalares OoO

- **Objetivo CPI < 1**
- Muchos procesadores superescalares permiten la ejecución fuera de orden (*OoO Processors, Out of Order Processors*).
- Las instrucciones se leen en orden, pero pueden ejecutarse en desorden.
- Las instrucciones se bloquean si sus operandos no están disponibles.
- Las instrucciones inician su ejecución cuando tiene sus operandos disponibles y la correspondiente U.F. está libre.
- La mayoría de los procesadores de propósito general actuales son superescalares con ejecución fuera de orden.



Intel Core 2 Architecture



VLIW: Very Long Instruction Word (Arquitecturas con tamaño de instrucción muy grande)

- Objetivo: **explotar ILP** (Instruction Level Parallelism), **CPI < 1**.
- Una instrucción especifica múltiples operaciones independientes
- Cada operación se ejecuta en una unidad funcional predeterminada
- La planificación de las instrucciones es estática: realizada por el compilador
- La planificación estática permite usar menos hardware de control y una arquitectura más simple:
  - Mayor frecuencia
  - Menor consumo
  - Menor coste
  - Más espacio para recursos (registros, UFs, caches, …)
  - Mayor facilidad de verificación
- El procesador sigue estando segmentado.
- Un porcentaje muy grande de los procesadores que se fabrican son embedded y de éstos muchos son VLIW.

Memoria de instrucciones

…

Instrucción

op1	op2	op3	op4	op5

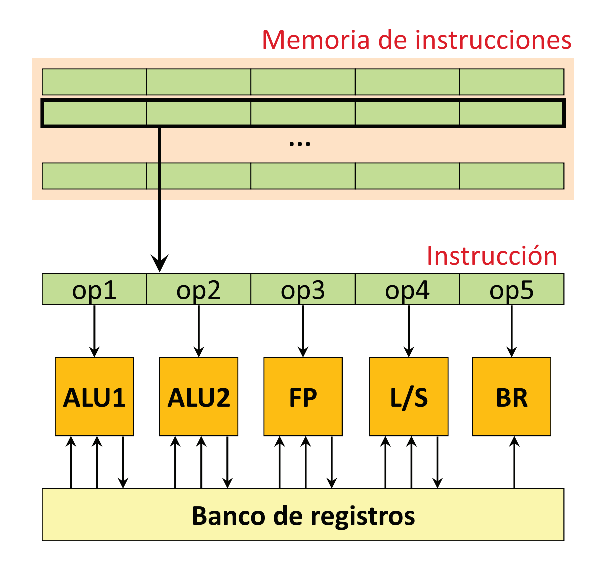
ALU1 ALU2 FP L/S BR

Banco de registros

VLIW vs Superescalar

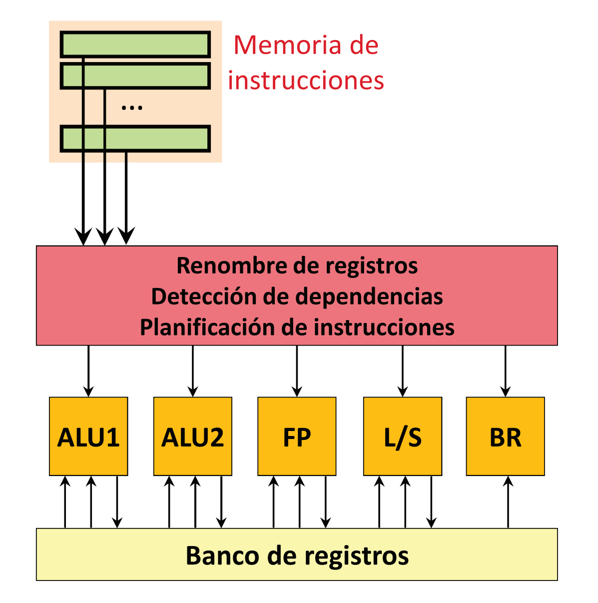
VLIW

- El compilador decide cuándo y dónde se ejecuta una operación

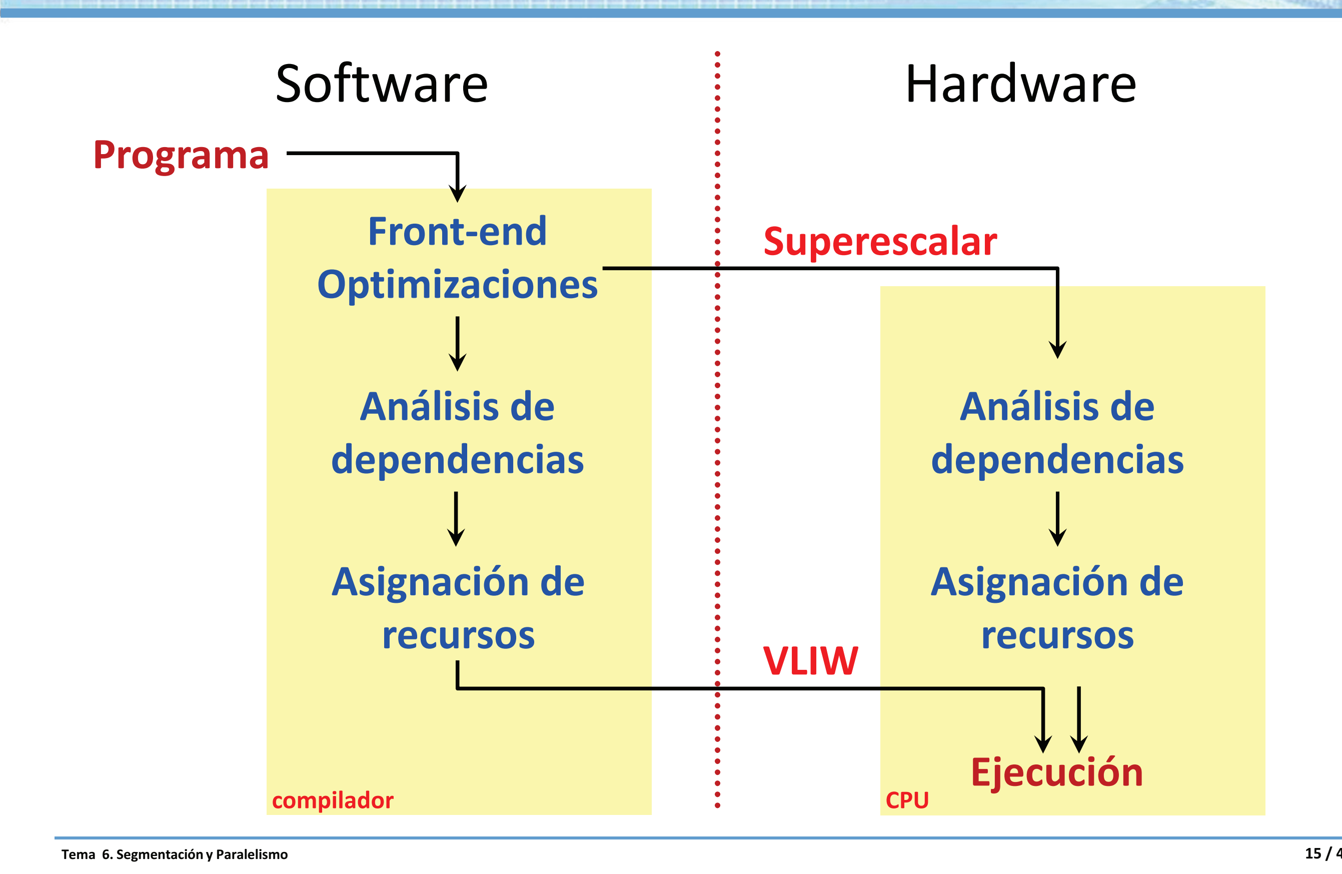


Superescalar

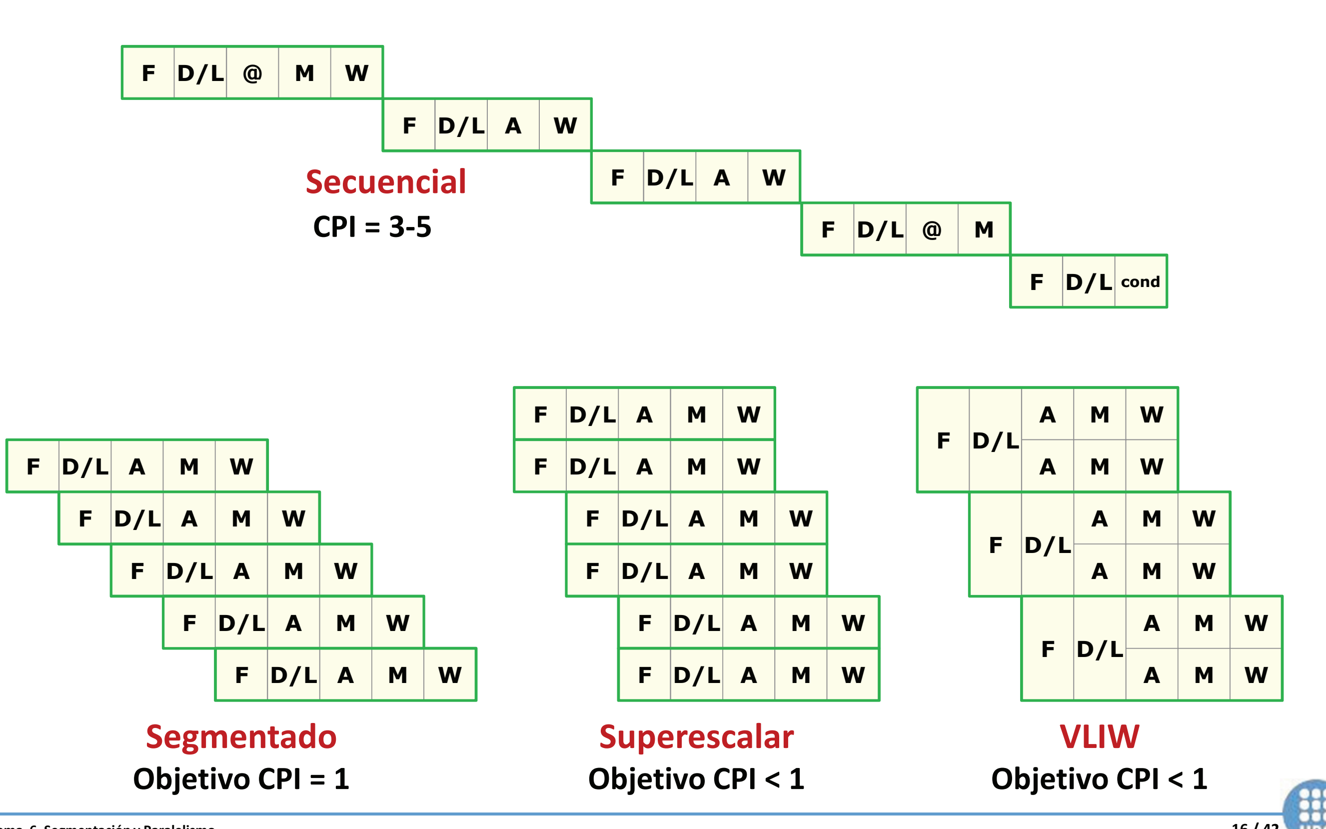
- El hardware decide



VLIW vs Superescalar



Resumiendo ILP

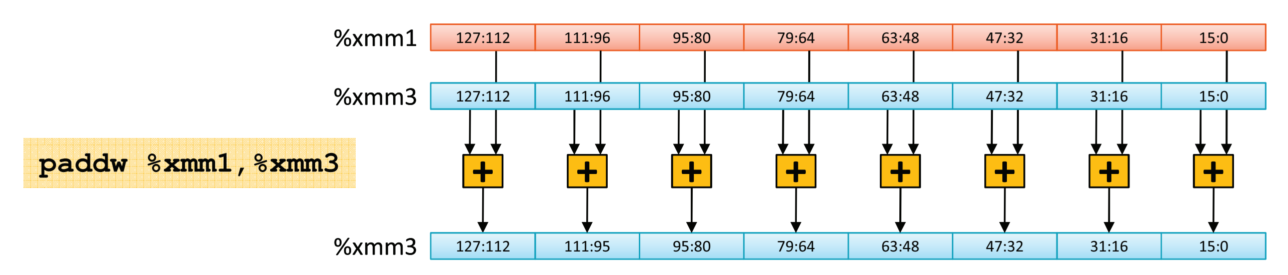


Índice

- **Introducción**
- **Paralelismo a Nivel de Instrucciones (ILP)**
- **Paralelismo a Nivel de Datos (DLP)**
  - SIMD
  - Procesadores Vectoriales
- **Paralelismo a Nivel de Thread (TLP)**
- **Ejemplos Reales**

SIMD (Single Instruction Multiple Data)

- 1 única instrucción que permite operar con múltiples datos del mismo tipo.



- Especialmente pensadas para aplicaciones multimedia: procesado de imagen, sonido, …
- La mayoría de los procesadores actuales tienen extensiones SIMD. En los procesadores x86 están disponibles desde hace años:
  - **MMX**, MultiMedia eXtension, Multiple Math eXtension, or Matrix Math eXtension, introducido en 1997 en los Pentium MMX
  - **SSE**, Streaming SIMD Extensions, introducido en 1999 en los Pentium III
  - **SSE2**, introducido en 2001 en los Pentium 4
  - **SSE3**, introducido en 2004 en los Pentium 4 Prescott (AMD lo amplió en 2005 en el Athlon)
  - **SSE4** , introducido en 2007 en los Intel Core y AMD K10
  - **SSE5**, anunciado por AMD en 2007, previsto para AMD Bulldozer en 2011
  - **AVX**, Advanced Vector eXtensions, anunciado por Intel en 2008, en el Sandy Bridge de 2011

Procesadores Vectoriales

- Los primeros supercomputadores (desde los años 70 hasta mediados de los 90) fueron procesadores vectoriales. **Ahora obsoletos**.
- Orientados al cálculo científico en coma flotante
  - Misma operación sobre distintos datos (vectores o matrices)
  - Muchas operaciones independientes
- En un procesador de propósito general las aplicaciones científicas tienen un rendimiento bastante limitado:

```
for (i=0; i<128; i++)
  A[i] = B[i]+C[i];
```

```
for: F1 ← B[R1]
     F2 ← C[R1]
     F8 ← F2+F1
     A[R1] ← F8
     R1 ← R1+8
     R9 ← R9-1
     BNE R9, for
```

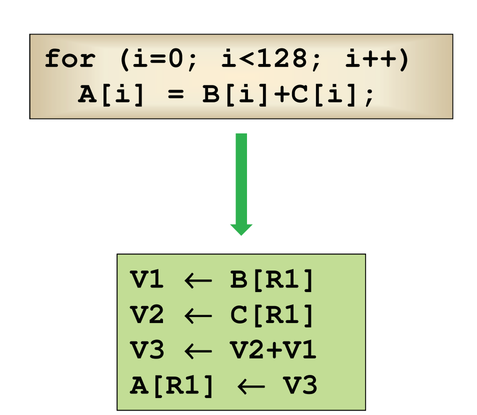
Desenrollar 2

```
for (i=0; i<128; i+=2){
  A[i]   = B[i]+C[i];
  A[i+1]= B[i+1]+C[i+1];
}
```

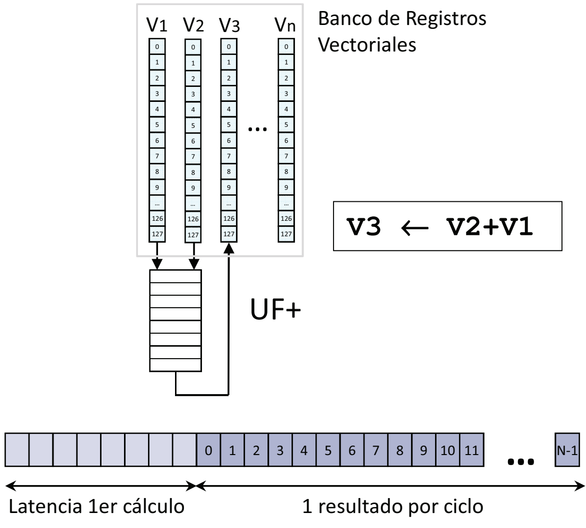
```
for: F1 ← B[R1]
     F2 ← C[R1]
     F3 ← B[R1+8]
     F4 ← C[R1+8]
     F8 ← F2+F1
     F9 ← F3+F4
     A[R1] ← F8
     A[R1+8] ← F8
     R1 ← R1+16
     R9 ← R9-2
     BNE R9, for
```


Procesadores Vectoriales

- Los procesadores vectoriales disponen de instrucciones que operan con vectores de números en coma flotante:
  - Usan **Registros Vectoriales** de tamaño fijo (**MVL** , p.e. 128 elementos).



- **Operaciones independientes**. Permite un elevado grado de segmentación de las Unidades Funcionales y tener tiempos de ciclo muy pequeños.



Cronograma de ejecución de una instrucción vectorial

Procesadores Vectoriales

- Acceso a datos muy eficiente. Acceso a memoria con patrones conocidos (localidad espacial). Acceso a registros vectoriales (localidad temporal).
- La memoria principal está especialmente organizada para soportar accesos a memoria con patrones regulares: acceso a elementos consecutivos en memoria (stride 1); acceso a elementos con distancia constante entre ellos (stride P).
- Los datos que utilizan las U.F. Vectoriales no pasan por Memoria Cache.
- Se reduce la sobrecarga debida al control de los bucles y los saltos condicionales.
- El rendimiento de estos procesadores está limitado por la fracción de código que se puede vectorizar (Ley de Amdahl).
- El compilador (vectorizador) es un elemento fundamental del sistema.

```
for (i=0; i<N; i++)
  A[i]=B[i]+C[i];
```

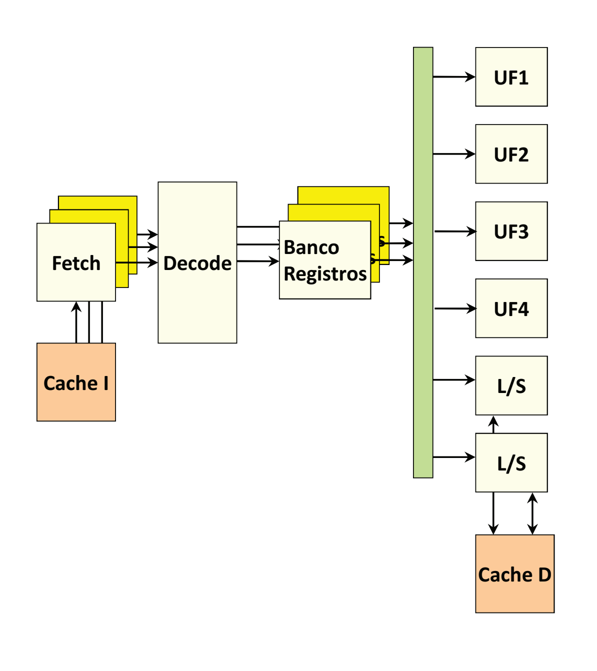
```
for (ii=0; ii<N; ii+=MVL)
  for (i=ii; i<min(ii+MVL,N); i++)
    A[i]=B[i]+C[i]; //N%MVL == 0
```

```
for: V1 ← B[R1] ;loadV
     V2 ← C[R1] ;loadV
     V3 ← V2+V1 ;addV
     A[R1] ← V3 ;storeV
     R1 ← R1+8*MVL
     R9 ← R9-MVL
     BNE R9, for
```


Índice

- **Introducción**
- **Paralelismo a Nivel de Instrucciones (ILP)**
- **Paralelismo a Nivel de Datos (DLP)**
- **Paralelismo a Nivel de Thread (TLP)**
  - Multithreading
  - Multiprocesadores
- **Ejemplos Reales**

Multithreading



- Un procesador actual dispone de múltiples unidades funcionales.
- Se intenta iniciar P instrucciones/operaciones por ciclo.
- Los programas no siempre disponen de suficiente ILP.
- Una forma de aprovechar el hardware disponible es intentar ejecutar instrucciones de threads diferentes.
- Sólo es necesario multiplicar alguno de los elementos hardware: Fetch y Banco de Registros. Hay que mantener el estado de cada thread en ejecución.
- El SO ve tantas CPUs lógicas como estados puede mantener la CPU.

Multithreading

Thread 0

A	B	C
D	E	
F	G	
H	I	J

Thread 1

A	B	
C	D	
E	F	G
H	I	

Block Multithreading

	A	B	C
Cache miss	D	E	
	A	B	
	C	D	
Cache miss	E	F	G
	F	G	
Cache miss	H	I	J
	H	I	

Interleaved Multithreading

A	B	C
A	B	
D	E	
C	D	
F	G	
E	F	G
H	I	J
H	I	

Simultaneous Multithreading

A	B	C
A	B	D
E	C	D
F	G	E
F	G	H
I	J	H
I		

Switch on Event Multithreading
Se cambia de thread cuando se produce un evento de alta latencia (oculta la latencia).

Fine Grained Multithreading
Se cambia de thread en cada ciclo. Puede haber múltiples threads activos.

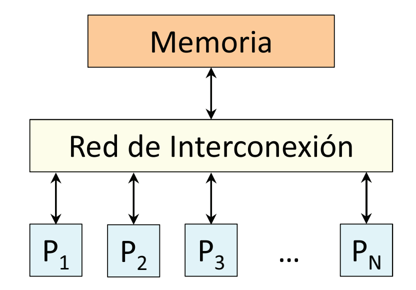
SMT (hyperthreading de Intel)
Se lanzan instrucciones de diferentes threads simultáneamente.
Encaja de forma natural con los superescalares.

Multiprocesadores

- Un multiprocesador es un computador que tiene N procesadores.
- En un mismo computador pueden ejecutarse varios threads pertenecientes a una misma aplicación o a aplicaciones independientes.
- Los sistemas multiprocesador pueden utilizarse para:
  - Ejecutar una aplicación paralela entre todos los elementos de proceso del computador. **El objetivo es la velocidad: Supercomputación.**
  - Ejecutar más aplicaciones por unidad de tiempo. **El objetivo es el throughput: Servidores de aplicaciones.**
  - O una mezcla de ambos.

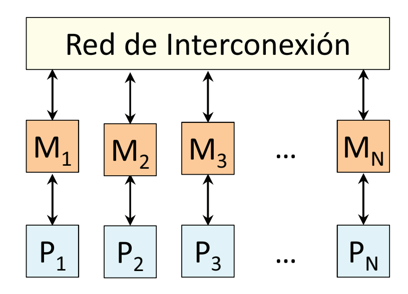
Multiprocesadores: Organización

- **Multiprocesadores con Memoria Compartida**



- Existe un único espacio de direcciones compartido por todos los procesadores.
- La red de interconexión permite a cualquier procesador acceder a cualquier posición de memoria.

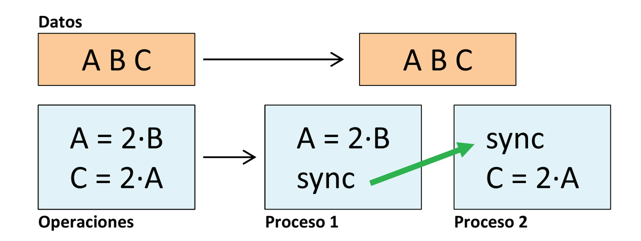
- **Multiprocesadores con Memoria Distribuida**



- Los procesadores sólo pueden acceder a su memoria local.
- La red de interconexión permite a cualquier procesador comunicarse con cualquiera de los procesadores del sistema.

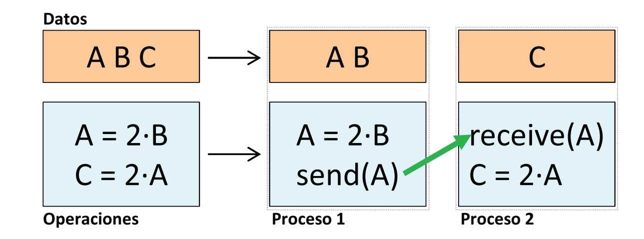
Multiprocesadores: Programación

- **Modelo de variables compartidas**



- Las operaciones se dividen en procesos.
- Los datos son compartidos por los procesos.
- Se requieren primitivas de sincronización:
  - Señalización y Acceso Exclusivo

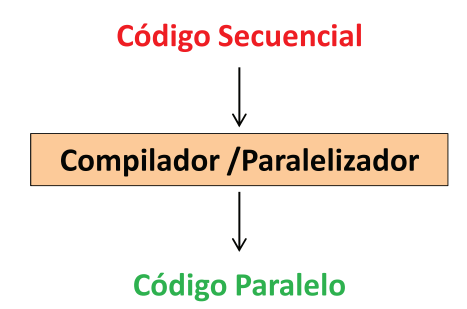
- **Modelo de paso de mensajes**



- Las operaciones y los datos se descomponen en procesos.
- Los procesos sólo tienen acceso a directo a los datos privados (locales).
- Los datos no locales se acceden mediante intercambio de mensajes entre procesos.

Multiprocesadores: Programación

- **Situación Ideal**



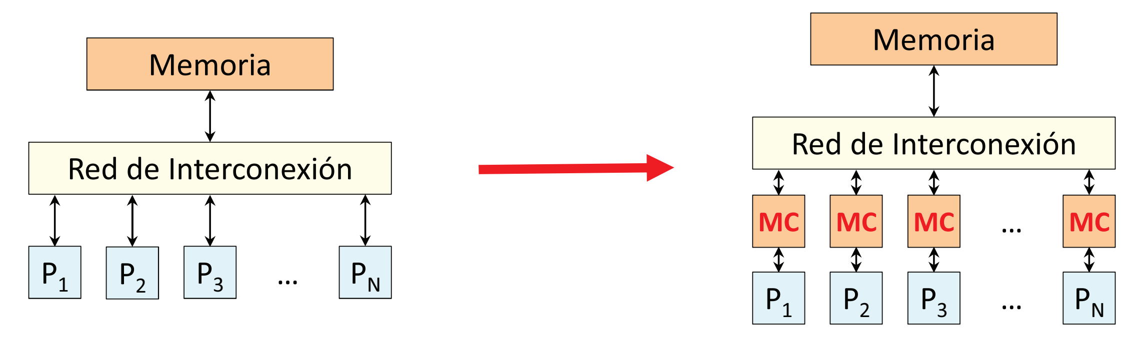
- Este es el modelo de programación ideal. Sin embargo, la tecnología de compilación actual no permite obtener buenos rendimientos en los sistemas multiprocesadores.

- **El modelo de programación y la organización**

		Modelo de Programación	
		Variables Compartidas	Paso de Mensajes
Organización	Memoria Compartida	Combinación natural Poco Escalable Programación fácil	Poco Escalable Programación difícil
	Memoria Distribuida	Programación fácil Escalable Implementación difícil	Combinación natural Escalable Programación difícil

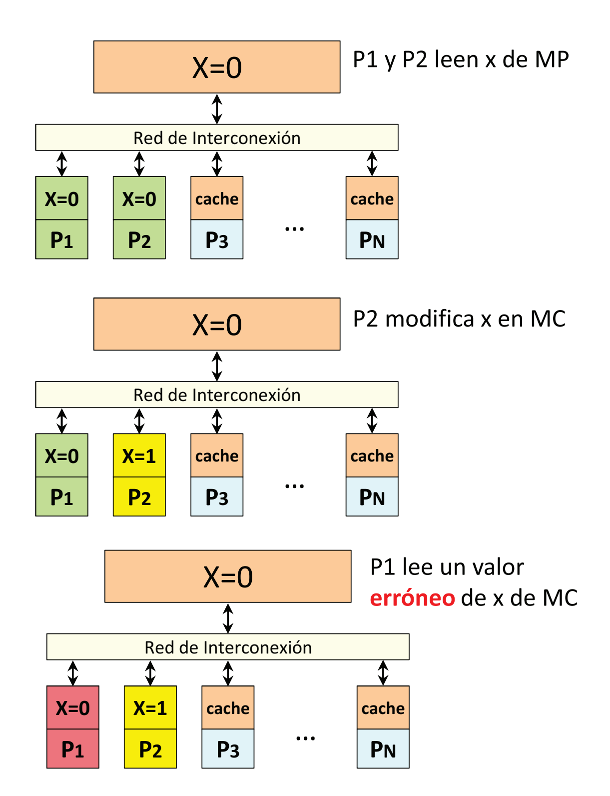
Multiprocesadores: Mejora de la Organización

- La red de interconexión aumenta la latencia con memoria.
- El uso de **memorias cache** locales de gran capacidad es **imprescindible**.



- El uso de memorias cache locales en un entorno de memoria compartida provoca la aparición de un problema que el hardware ha de resolver: **COHERENCIA de MEMORIA**.
- Un sistema de memoria es coherente si cualquier lectura de un dato devuelve el último valor escrito sobre esa posición de memoria. Existen 2 temas a tener en cuenta:
  - Coherencia
  - Consistencia

El Problema de la Coherencia de Memoria



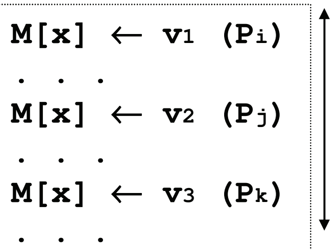
- Un sistema es coherente si cumple tres condiciones:

```
M[x] ← v (Pi)
. . .
w ← M[x] (Pi)
```

Ninguna escritura en M[x]
⇒ w = v

```
M[x] ← v (Pi)
. . .
w ← M[x] (Pj)
```

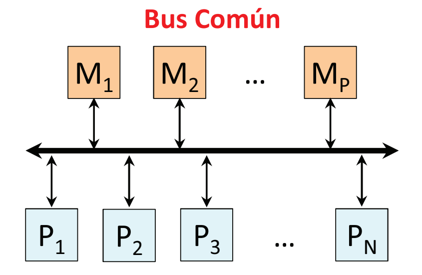
Ninguna escritura en M[x] tiempo suficiente para que la escritura sea visible
⇒ w = v

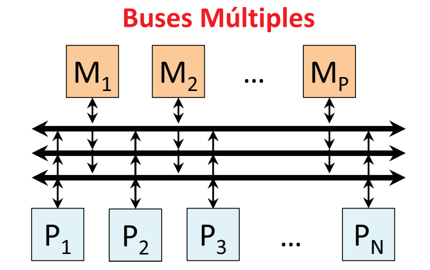


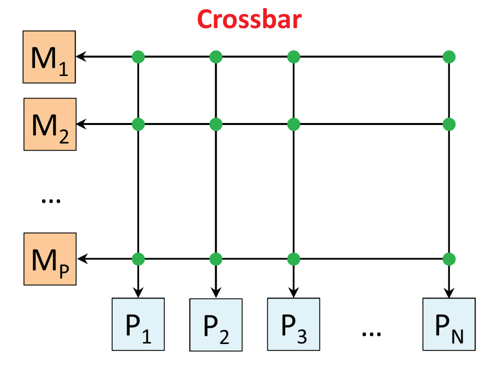
Todas las escrituras se ven en el mismo orden por todos los procesadores (M[x] = v_1, v_2, v_3). Nunca puede ocurrir que un procesador vea : M[x] = v_1, v_3, v_2

Redes de Interconexión

- Elemento fundamental en el rendimiento de un multiprocesador







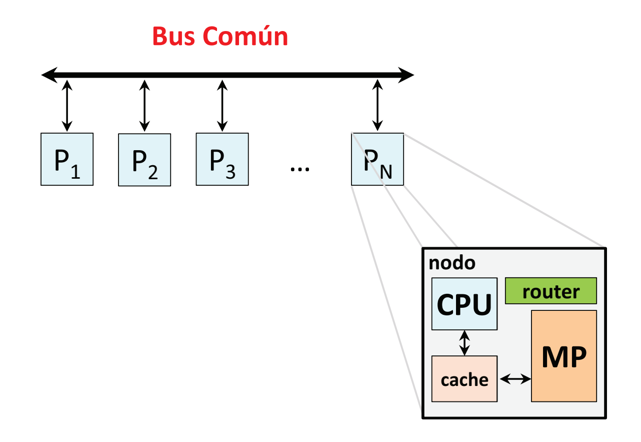
Bus Común
- Barato
- Ancho de banda bajo

Crossbar
- Caro (para muchos procesadores)
- Ancho de banda alto

Buses Múltiples
- Compromiso entre bus común y crossbar.

Redes de Interconexión

- La red de interconexión también es fundamental en los multiprocesadores con memoria distribuida



Buses Múltiples

P1 P2 P3 … PN

Crossbar

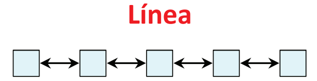
P1 P2 P3 … PN

Redes de Acceso a Memoria Uniforme (UMA)
- Cualquier pareja de nodos se comunican con igual coste de comunicación.
- Redes poco escalables.

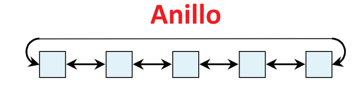
Redes de Interconexión

- Los multiprocesadores con memoria distribuida pueden utilizar conexiones punto a punto.

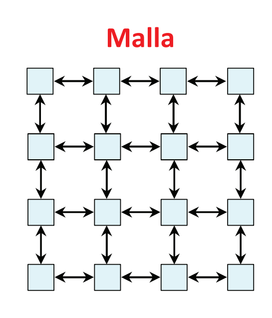
Línea



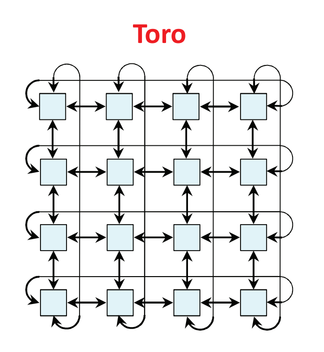
Anillo



Malla



Toro

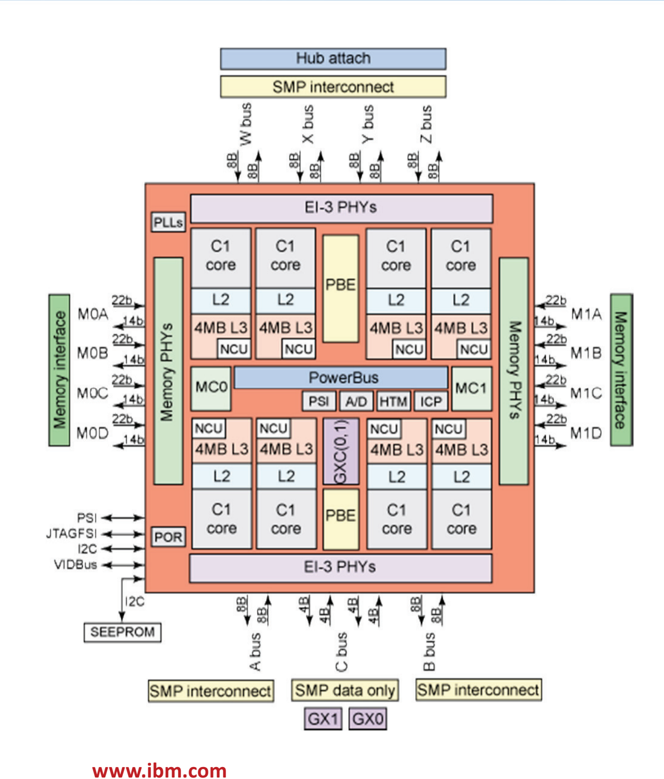


Redes de Acceso a Memoria No Uniforme (NUMA)
- El coste de la comunicación entre dos nodos depende de la posición relativa de los nodos en la red.
- Redes Escalables.

Índice

- Introducción
- Paralelismo a Nivel de Instrucciones (ILP)
- Paralelismo a Nivel de Datos (DLP)
- Paralelismo a Nivel de Thread (TLP)
- Ejemplos Reales
  - Un procesador de propósito general
  - Un TV 3D
  - Una GPU
  - Una consola
  - Supercomputador MareNostrum

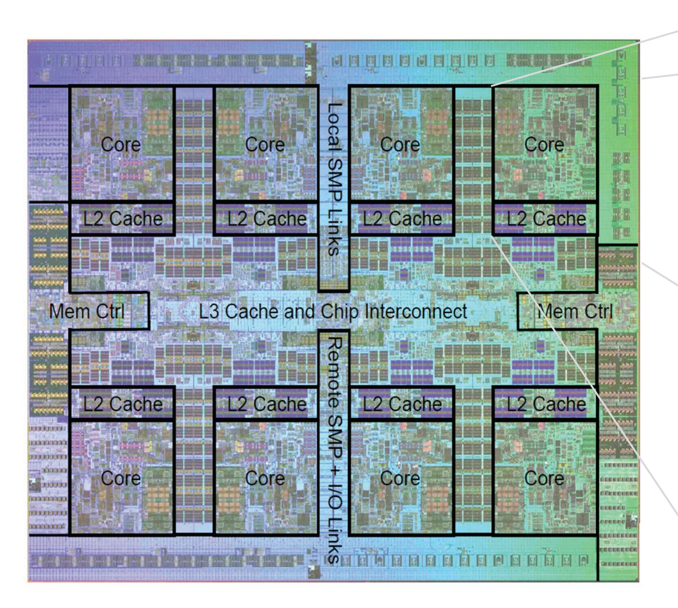
IBM Power 7

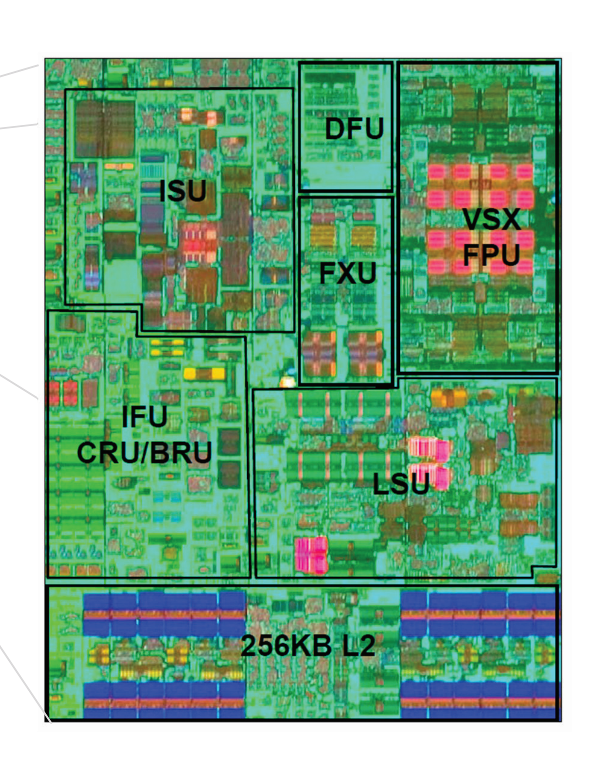


www.ibm.com

- Frecuencia: 3 - 4.25 GHz
- 8 cores por chip, 4 SMT threads por core
- Ejecución fuera de orden
- 12 U.F.s por core (2 L/S, 4 CF, 1 Vector, …)
- Altivec Vector SIMD instructions
- 32KB + 32KB L1 por core
- 256 KB L2 por core
- 32 MB L3 (eDRAM con ECC), compartida con protocolo de coherencia incluido (hasta 8 chips)
- 2 controladores de memoria, 8 canales, 100 GB/s
- Todos los buses de comunicación protegidos con ECC
- Rendimiento de pico
  - 32,12 GFLOPS por core
  - 264.96 GFLOPS por chip
- Gestión muy eficiente del consumo
  - ON/OFF de los cores dinámicamente
  - Variación dinámica de la frecuencia por core

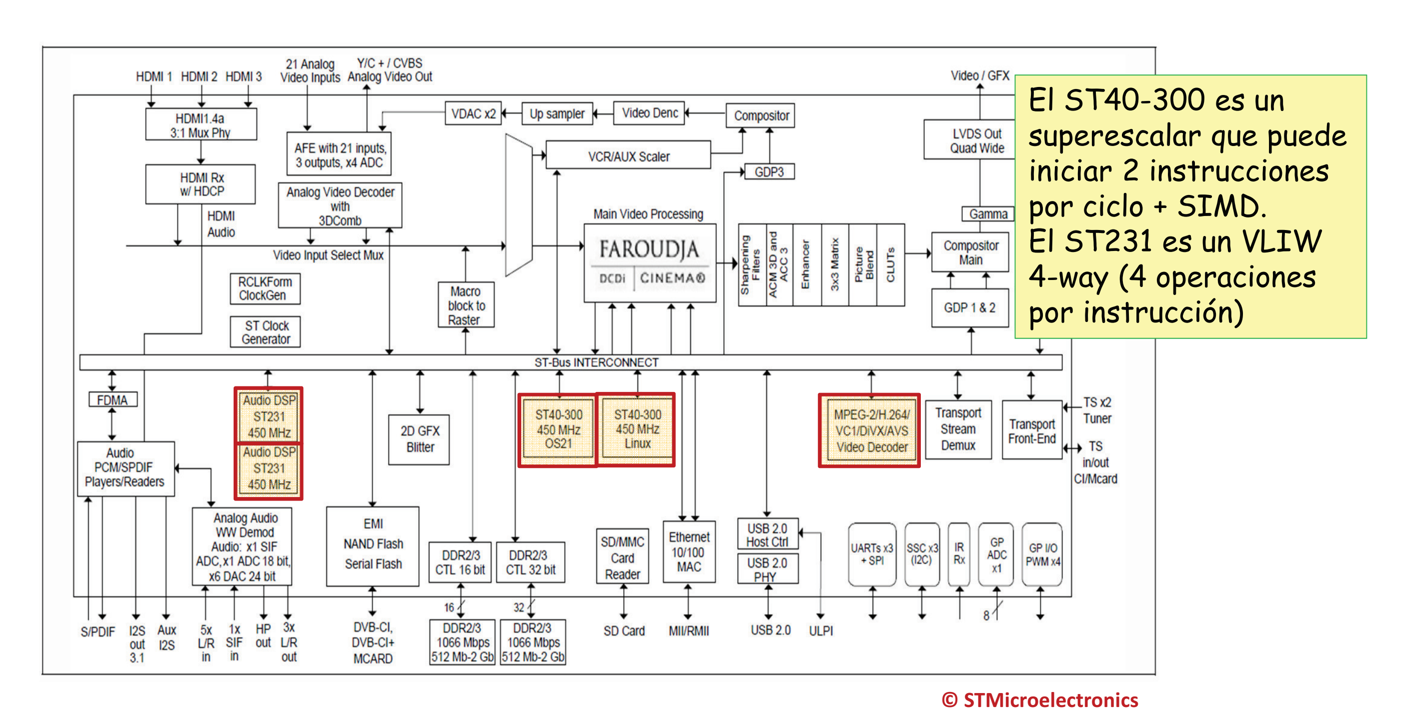
IBM Power 7





www.ibm.com

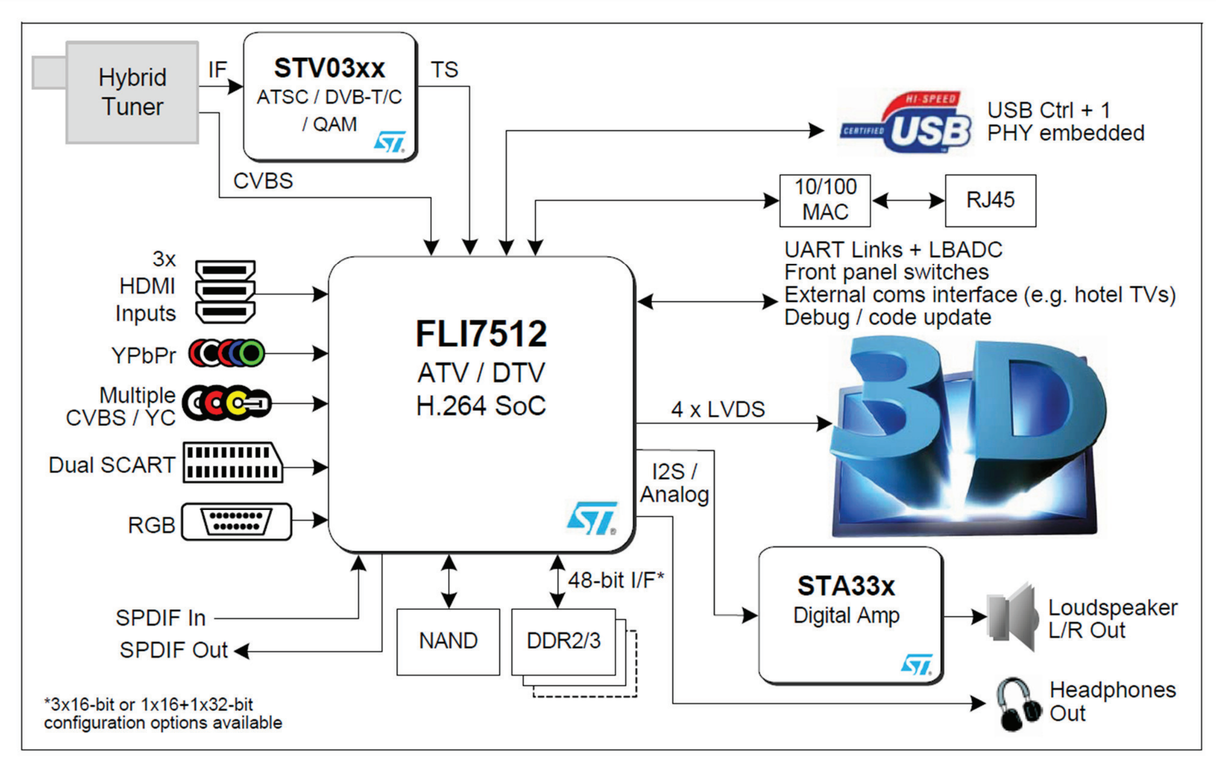
ST FLI7521 → TV 3D



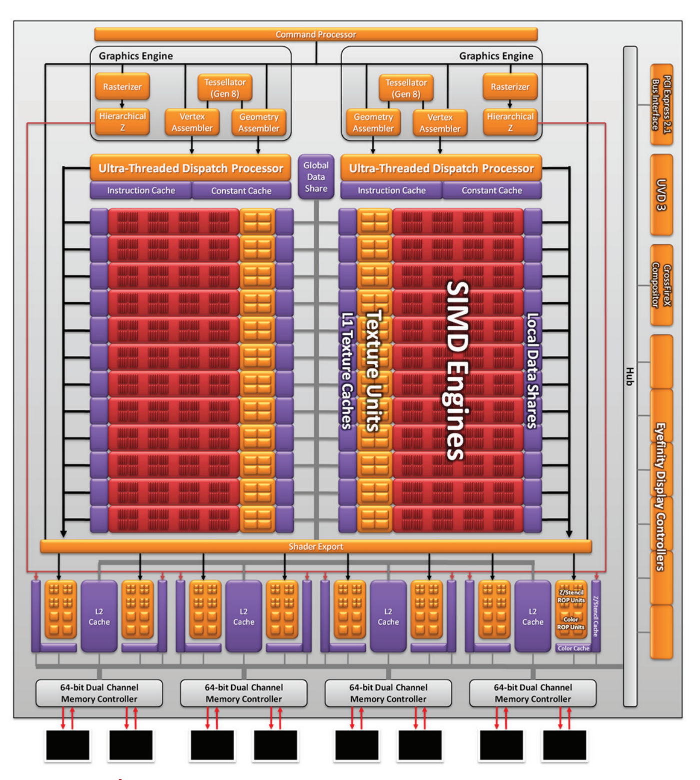
El ST40-300 es un superescalar que puede iniciar 2 instrucciones por ciclo + SIMD.
El ST231 es un VLIW 4-way (4 operaciones por instrucción)

Algunos de los elementos específicos también son CPUs.

ST FLI7521 → TV 3D



AMD GPU

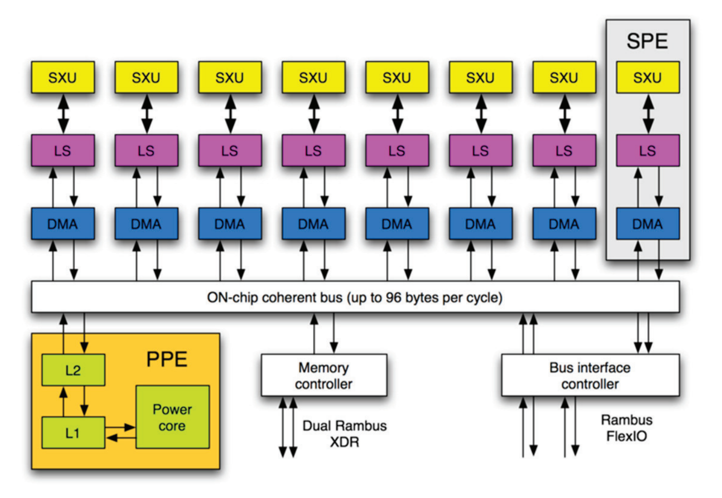


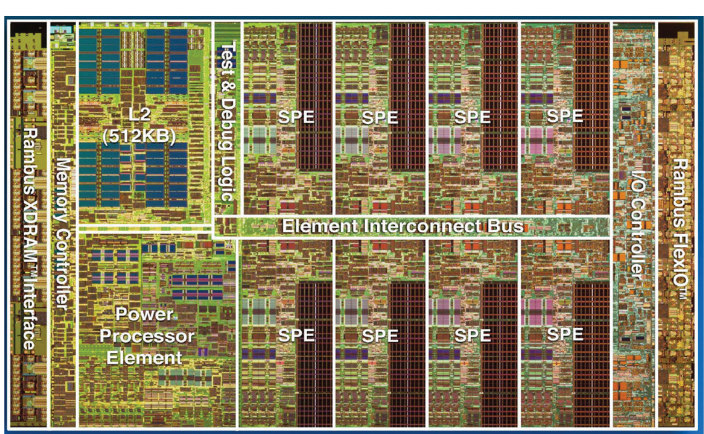
www.amd.com

AMD Radeon™ HD Serie 6970

- Multi-core
- Multi-thread
- VLIW
- SIMD
- Frecuencia de la GPU: 880 MHz
- 2 GB de memoria GDDR5
- Frecuencia de memoria 1375 MHz
- 8 canales con memoria
- 176 GB/s de acho de banda de memoria
- 2,7 TFLOPs en simple precisión
- 683 GFLOPs es doble precisión
- 1536 elementos de cálculo

Cell, procesador gráfico de la PS3





- Multi-thread, Multi-core, Memoria Distribuida
- 1 PPE (Power Processing Element) . CPU de propósito general. Encargado de distribuir la carga entre los SPEs y recoger los resultados. Es un PowerPC.
- 8 SPE (Synergistic Processing Elements). Procesan las tareas que le encarga el PPE. La comunicación entre ellos es vía DMA. No disponen de Memoria cache. Sólo pueden acceder a su Memoria RAM Local. Dispone de unidades de cálculo vectorial del tipo SIMD.

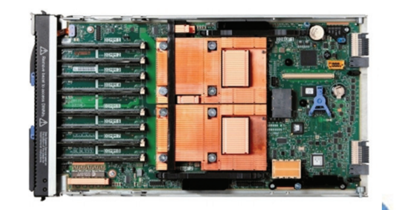
Cell, procesador gráfico de la PS3



- Arquitectura diseñada para grandes cargas de cálculo de aplicaciones digitales: películas, videojuegos y cálculo numérico.
- Las tareas que corren en los SPEs han de estar vectorizadas
- Toda la coordinación entre las tareas la ha de realizar el PPE.
- Programar este procesador no es trivial, como tampoco lo era programar el Emotion Engine (procesador gráfico de la PS2).



- PowerXCell es una variante del Cell. El PowerXCell 8i está disponible en los Servidores Blade QS22 de IBM.
- El PowerXCell 8i a 3.2 GHz tiene una potencia de pico de 204.8 GFLOPS en doble precisión.
- El #7 del top500 (Nov-2010) utiliza PowerXCell 8i.



- Los TV 3D de gama alta de Toshiba utilizan Cell.
- Proyecto de STI (Sony, Toshiba, IBM), 400 ingenieros trabajando durante 4 años con un presupuesto de 2000 millones de €.

Supercomputador MareNostrum



Visita guiada al BSC en la próxima clase de teoría.